# CHAPTER 3 SELECTIONS

MOTIVATION

- If you assigned a negative value for radius in πr^2, the program would print an invalid result. If the radius is negative, you don't want the program to compute the area. How can you deal with this situation?

- Say you run a casino, how would you determine if a slot machine yielded a win or a loss?

- You work for the government, how might you decide if a person is eligible for a tax refund or not?

- Moreover, how would you make a program do something over-and-over, like maintain a set of accounts for a bank?
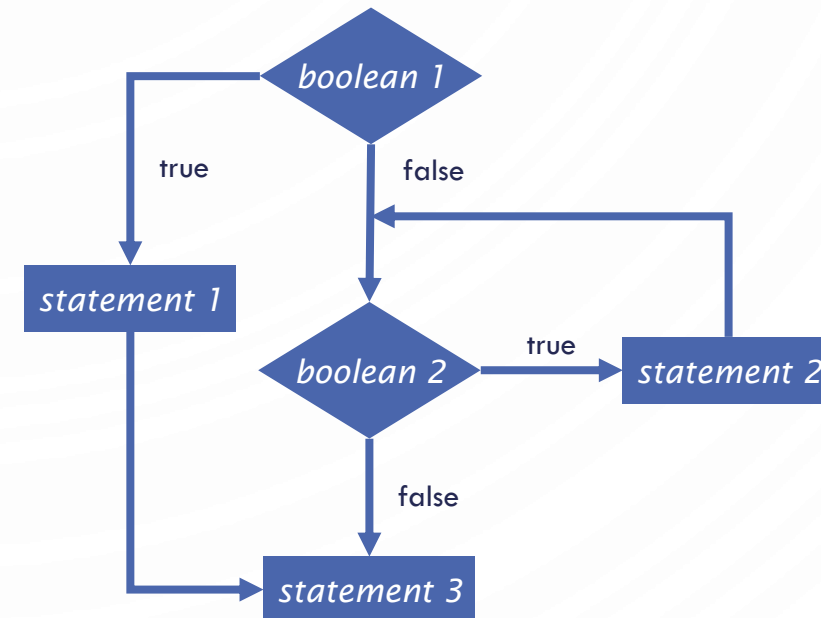
CONTROL FLOW

- Control flow.
  - Sequence of statements that are actually executed in a program.
  - Conditionals and loops: enable us to choreograph control flow.
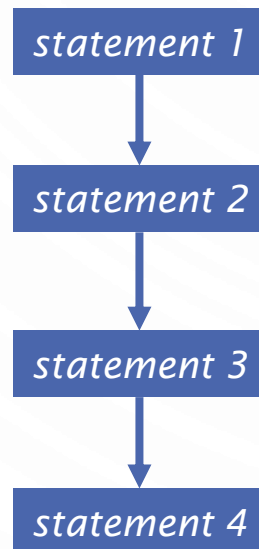
straight-line control flow

control flow with conditionals and loops

BOOLEANS

BOOLEAN OPERATORS

A AND B   A OR B   A NOT B
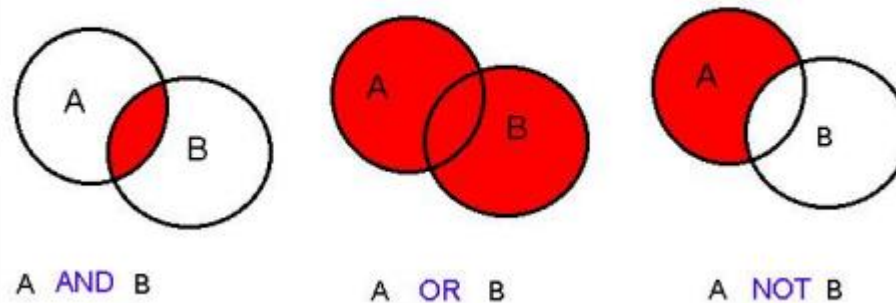
BOOLEAN HAIR LOGIC

A   B

AND   OR   XOR
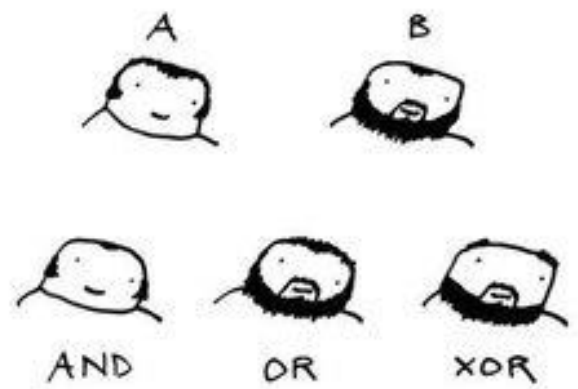
THE BOOLEAN TYPE AND OPERATORS

- Often in a program you need to compare two values, such as whether i is greater than j. Java provides six comparison operators (also known as relational operators) that can be used to compare two values. The result of the comparison is a **Boolean value**: true or false.

- `boolean b = (1 > 2);`

RELATIONAL OPERATORS

Java Operator	Mathematics Symbol	Name	Example (radius is 5)	Result
<	<	less than	radius < 0	false
<=	≤	less than or equal to	radius <= 0	false
>	>	greater than	radius > 0	true
>=	≥	greater than or equal to	radius >= 0	true
==	=	equal to	radius == 0	false
!=	≠	not equal to	radius != 0	true

This is what you write in pseudocode!

LOGICAL OPERATORS

Java Operator	Math Symbol	Name	Description
!	¬	not	logical negation
&&	∧	and	logical conjunction
||	∨	or	logical disjunction
^	⊕ or v̲	exclusive or	logical exclusion

This is what you write in pseudocode!

TRUTH TABLE FOR OPERATOR ¬

p	$\neg p$	Example (assume age = 24, weight = 140)
true	false	$\neg(age > 18)$ is false, because $(age > 18)$ is true.
false	true	$\neg(weight = 150)$ is true, because $(weight = 150)$ is false.

TRUTH TABLE FOR OPERATOR ∧

p_1	p_2	$p_1 \wedge p_2$	Example (assume age = 24, weight = 140)
false	**false**	false	$(age \leq 18) \wedge (weight < 140)$ is false
false	**true**	false	$(age \leq 18) \wedge (weight \leq 140)$ is false
true	**false**	false	$(age > 18) \wedge (weight < 140)$ is false
true	**true**	true	$(age > 18) \wedge (weight \leq 140)$ is true

TRUTH TABLE FOR OPERATOR ∨

p_1	p_2	$p_1 \lor p_2$	Example (assume age = 24, weight = 140)
false	**false**	false	$(age \leq 18) \lor (weight < 140)$ is false
false	**true**	true	$(age \leq 18) \lor (weight \leq 140)$ is true
true	**false**	true	$(age > 18) \lor (weight < 140)$ is true
true	**true**	true	$(age > 18) \lor (weight \leq 140)$ is true

EXERCISE

- Draw the truth table for exclusive or ($\oplus$)

TRUTH TABLE FOR OPERATOR $\oplus$

p_1	p_2	$p_1 \oplus p_2$	Example (assume age = 24, weight = 140)
false	**false**	false	$(age \leq 18) \oplus (weight < 140)$ is false
false	**true**	true	$(age \leq 18) \oplus (weight \leq 140)$ is true
true	**false**	true	$(age > 18) \oplus (weight < 140)$ is true
true	**true**	false	$(age > 18) \oplus (weight \leq 140)$ is false

EXAMPLE PROGRAM

- Here is a program that check division by 2 and 3, 2 or 3, and 2 or 3 exclusive

```
1. public class TestBoolean {
2.    public static void main(String[] args) {
3.        Scanner s = new Scanner(System.in);
4.        System.out.print("Enter a number: ");
5.        int x = s.nextInt();
6.
7.        boolean divBy2 = x % 2 == 0;
8.        boolean divBy3 = x % 3 == 0;
9.
10.     System.out.println(x + " divisible by 2 and 3: " + divBy2 && divBy3);
11.     System.out.println(x + " divisible by 2 and 3: " + divBy2 || divBy3);
12.     System.out.println(x + " divisible by 2 and 3: " + divBy2  ^ divBy3);
13.  }
14. }
```

Recall you can make combined statements like: `(x % 2 == 0) && (x % 3 == 0)`

EXERCISE

- Let a user enter a year, and output whether or not it is a leap year. A year is a leap year if it is
  - Divisible by 4 but not by 100
  - OR
  - Divisible by 400

OPERATOR PRECEDENCE

1. () (expressions in parenthesis)
2. var++, var--
3. +, - (Unary plus and minus), ++var,--var
4. (type) Casting
5. ! (Not)
6. *, /, % (Multiplication, division, and remainder)
7. +, - (Binary addition and subtraction)
8. <, <=, >, >= (Relational operators)
9. ==, !=; (Equality)
10. ^ (Exclusive OR)
11. && (Conditional AND) Short-circuit AND
12. || (Conditional OR) Short-circuit OR
13. =, +=, -=, *=, /=, %= (Assignment operator)

OPERATOR PRECEDENCE AND ASSOCIATIVITY

- The expression in the parentheses is evaluated first. (Parentheses can be nested, in which case the expression in the inner parentheses is executed first.) When evaluating an expression without parentheses, the operators are applied according to the precedence rule and the associativity rule.

- If operators with the same precedence are next to each other, their associativity determines the order of evaluation. All binary operators except assignment operators are left-associative.

EXAMPLE

- Applying the operator precedence and associativity rule, the expression 3 + 4 * 4 > 5 * (4 + 3) - 1 is evaluated as follows:

```
3 + 4 * 4 > 5 * (4 + 3) - 1
                     ↑_____ (1) inside parentheses first
3 + 4 * 4 > 5 * 7 - 1
        ↑_____ (2) multiplication
3 + 16 > 5 * 7 - 1
            ↑_____ (3) multiplication
3 + 16 > 35 - 1
   ↑_____ (4) addition
19 > 35 - 1
        ↑_____ (5) subtraction
19 > 34
   ↑_____ (6) greater than
false
```

SELECTIONS
AKA CONDITIONALS

CONDITIONALS IN PSEUDOCODE

1. `if` *booleanCondition* `then`

2.   Perform some operation

1. `if` *booleanCondition* `then`

2.   Perform some operation

3. `else`

4.   Perform some other operation

1. `if` *booleanCondition* `then`

2.   Perform some operation

3. `else if` *booleanCondition2* `then`

4.   Perform some other operation

5. …

6. `else`

7.   Perform final option of operation

ONE-WAY IF STATEMENTS IN JAVA

```
1. if (boolean-expression) {
2.    statement(s);
3. }
```
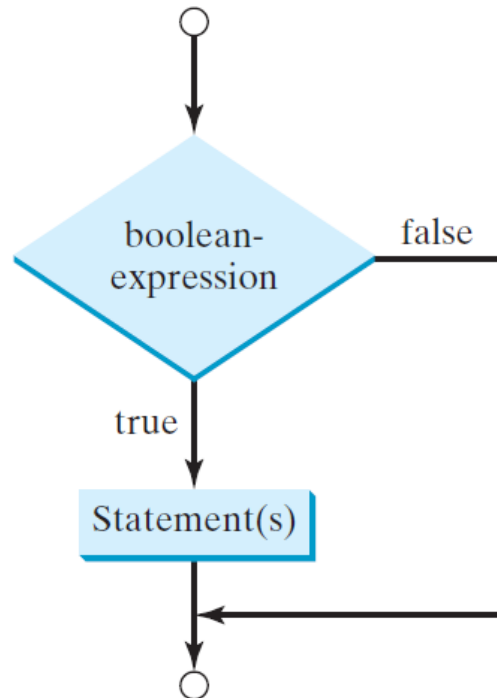


```
1. if (radius >= 0) {
2.    area = radius * radius * PI;
3.    System.out.println("The area"
4.       + " for the circle of radius "
5.       + radius + " is " + area);
6. }
```
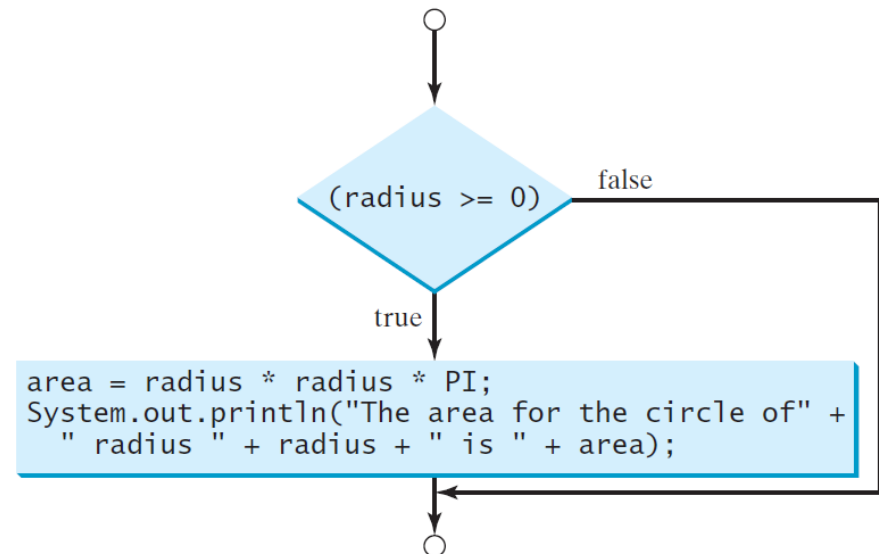
NOTE

- Parenthesis are required!

```
if i > 0 {
   System.out.println("i is positive");
}
```
(a) Wrong

```
if (i > 0) {
   System.out.println("i is positive");
}
```
(b) Correct

- Curly braces are optional ONLY FOR single statement blocks

```
if (i > 0) {
   System.out.println("i is positive");
}
```
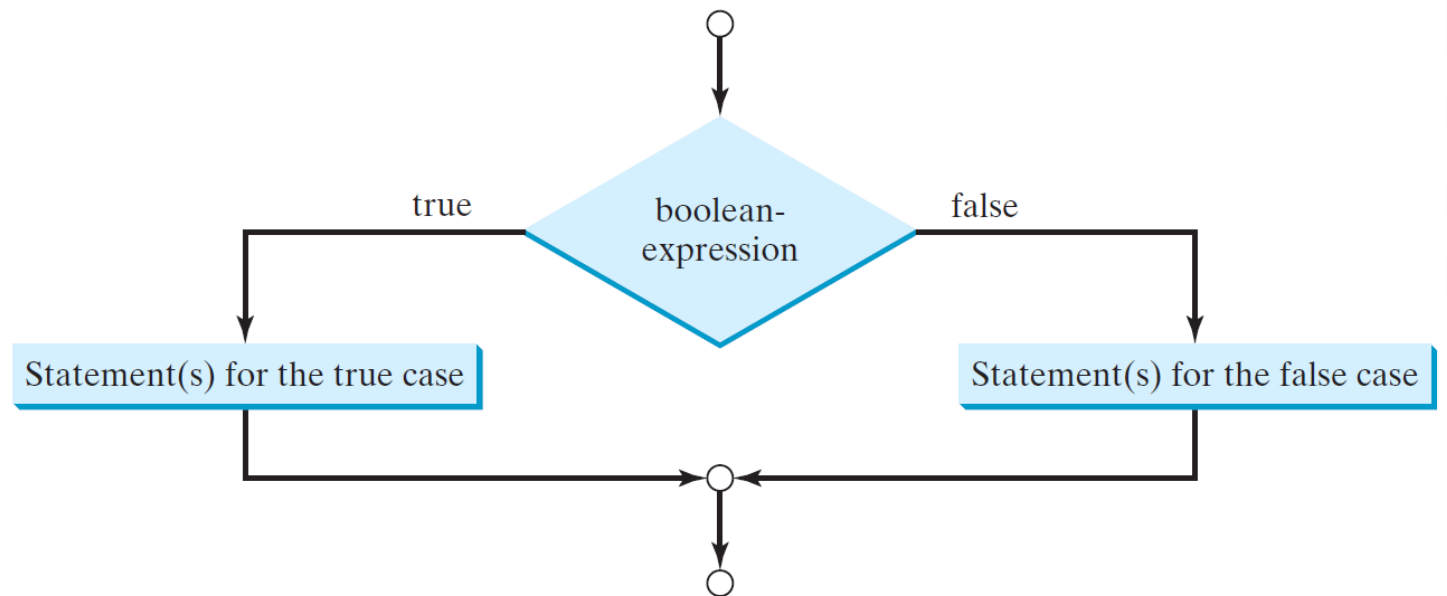(a)

Equivalent

```
if (i > 0)
   System.out.println("i is positive");
```
(b)

THE TWO-WAY IF STATEMENT

```
1.  if (boolean-expression) {
2.     statement(s)-for-the-true-case;
3.  }
4.  else {
5.     statement(s)-for-the-false-case;
6.  }
```

IF-ELSE EXAMPLE

```java
1. if (radius >= 0) {
2.    area = radius * radius * 3.14159;
3.    System.out.println("The area for the "
4.      + "circle of radius " + radius +
5.      " is " + area);
6. }
7. else {
8.    System.out.println("Negative input");
9. }
```

EXERCISE: GUESSING GAME

- Use `Math.random()` to generate a random number between 1 and 100:

  `(int)(Math.random()*99 + 1);`

- Have a user guess the number. If the user is correct output a winning message, otherwise output a losing message

- Lets solve together. Program along with me.

MULTIPLE ALTERNATIVE IF STATEMENTS

Note the syntax for else-if statements

```
if (score >= 90.0)
  System.out.print("A");
else
  if (score >= 80.0)
    System.out.print("B");
  else
    if (score >= 70.0)
      System.out.print("C");
    else
      if (score >= 60.0)
        System.out.print("D");
      else
        System.out.print("F");
```

(a)

Equivalent

```
if (score >= 90.0)
    System.out.print("A");
else if (score >= 80.0)
    System.out.print("B");
else if (score >= 70.0)
    System.out.print("C");
else if (score >= 60.0)
    System.out.print("D");
else
    System.out.print("F");
```
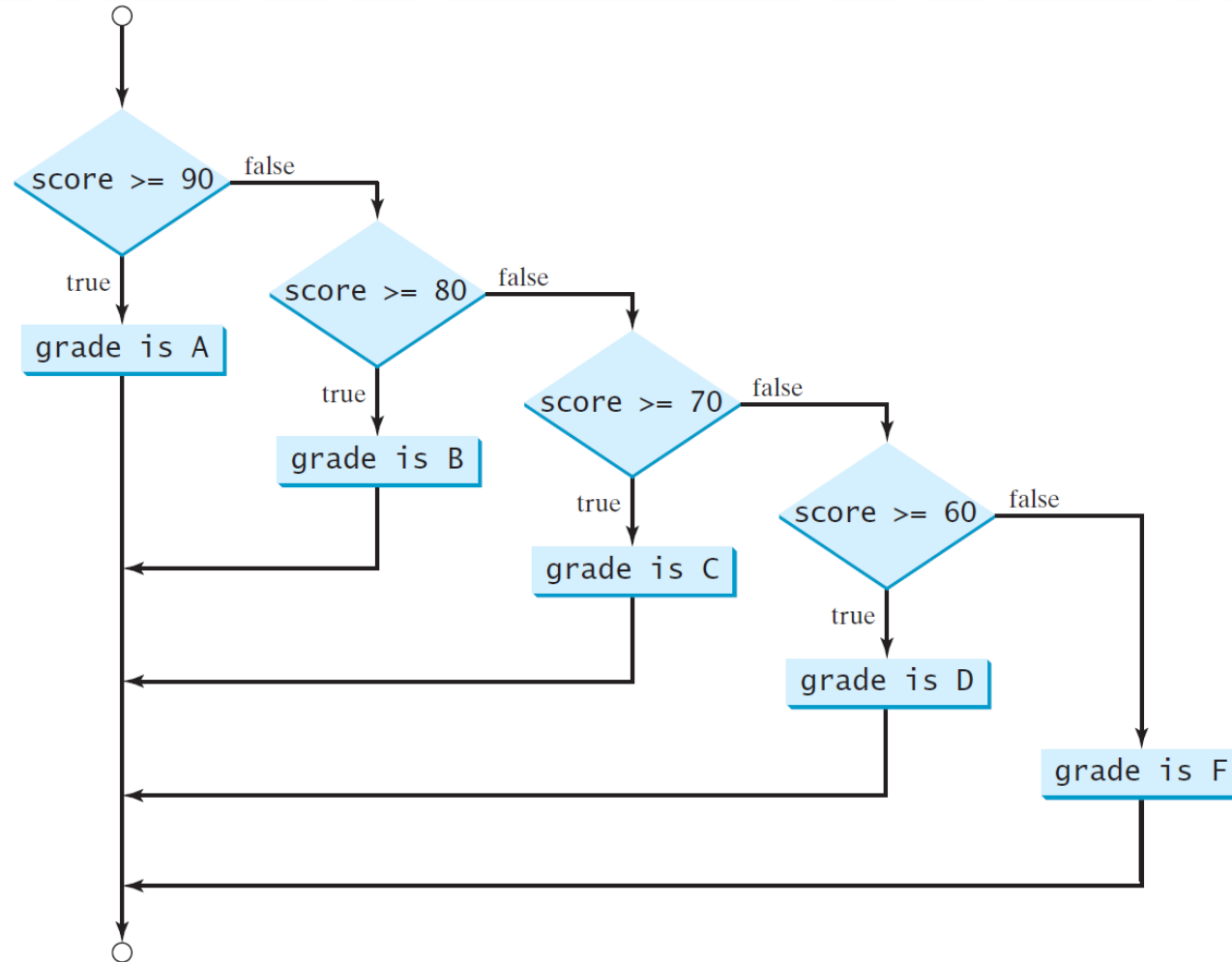
(b)

This is better

MULTI-WAY IF-ELSE STATEMENTS

TRACE IF-ELSE STATEMENT

Suppose score is 72.3

Condition is false

```
1.  if (score >= 90.0)
2.      System.out.print("A");
3.  else if (score >= 80.0)
4.      System.out.print("B");
5.  else if (score >= 70.0)
6.      System.out.print("C");
7.  else if (score >= 60.0)
8.      System.out.print("D");
9.  else
10.     System.out.print("F");
```

TRACE IF-ELSE STATEMENT

Suppose score is 72.3

```
1.    if (score >= 90.0)
2.       System.out.print("A");
3.    else if (score >= 80.0)
4.       System.out.print("B");
5.    else if (score >= 70.0)
6.       System.out.print("C");
7.    else if (score >= 60.0)
8.       System.out.print("D");
9.    else
10.      System.out.print("F");
```

Condition is false

TRACE IF-ELSE STATEMENT

Suppose score is 72.3

```java
1.    if (score >= 90.0)
2.       System.out.print("A");
3.    else if (score >= 80.0)
4.       System.out.print("B");
5.    else if (score >= 70.0)
6.       System.out.print("C");
7.    else if (score >= 60.0)
8.       System.out.print("D");
9.    else
10.      System.out.print("F");
```

Condition is true

TRACE IF-ELSE STATEMENT

Suppose score is 72.3

```
1.    if (score >= 90.0)
2.       System.out.print("A");
3.    else if (score >= 80.0)
4.       System.out.print("B");
5.    else if (score >= 70.0)
6.       System.out.print("C");
7.    else if (score >= 60.0)
8.       System.out.print("D");
9.    else
10.      System.out.print("F");
```

Output "C"

TRACE IF-ELSE STATEMENT

Suppose score is 72.3

```
1.    if (score >= 90.0)
2.        System.out.print("A");
3.    else if (score >= 80.0)
4.        System.out.print("B");
5.    else if (score >= 70.0)
6.        System.out.print("C");
7.    else if (score >= 60.0)
8.        System.out.print("D");
9.    else
10.       System.out.print("F");
```

Exit the block

NOTE

- The else clause matches the most recent if clause in the same block.

```
int i = 1, j = 2, k = 3;

if (i > j)
    if (i > k)
        System.out.println("A");
else
        System.out.println("B");
```
(a)

Equivalent

This is better
with correct
indentation

```
int i = 1, j = 2, k = 3;

if (i > j)
    if (i > k)
        System.out.println("A");
    else
        System.out.println("B");
```
(b)

NOTE, CONT.

- Nothing is printed from the preceding statement. To force the else clause to match the first if clause, you must add a pair of curly braces:

```java
1.   int i = 1, j = 2, k = 3;
2.
3.   if (i > j) {
4.     if (i > k)
5.         System.out.println("A");
6.   }
7.   else
8.       System.out.println("B");
```

- This statement prints B.

COMMON ERRORS

- Adding a semicolon at the end of an if clause is a common mistake.

```
1. if (radius >= 0);        ← Wrong!
2. {
3.    area = radius*radius*PI;
4.    System.out.println(
5.      "The area for the circle of radius " +
6.      radius + " is " + area);
7. }
```

- This mistake is hard to find, because it is not a compilation error or a runtime error, it is a logic error.

- This error often occurs when you use the next-line block style.

TIP

```
if (number % 2 == 0)
    even = true;
else
    even = false;
```

(a)

Equivalent

```
boolean even
    = number % 2 == 0;
```

(b)

Hint 1: If you **LIKE** getting big points off of style. I recommend writing (a)!

Hint 2: Hint 1 is sarcasm…

CAUTION

```
if (even == true)
   System.out.println(
      "It is even.");
```

(a)

Equivalent

```
if (even)
   System.out.println(
      "It is even.");
```
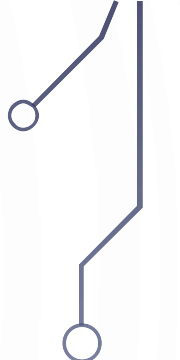
(b)

Hint 1: If you **LIKE** getting big points off of style. I recommend writing (a)!

Hint 2: Hint 1 is sarcasm…

EXERCISE: GUESSING GAME

- Extend the previous guessing game example to allow two guesses, and descriptive messages of a guess being correct, over, or under.

  - Hint: We have 3 conditions per guess…

- Lets solve together. Program along with me.

ADVANCED CONDITIONALS

PROBLEM: COMPUTING TAXES

- The US federal personal income tax is calculated based on the filing status and taxable income. There are four filing statuses: single filers, married filing jointly, married filing separately, and head of household. The tax rates for

| Marginal Tax Rate	Single	Married Filing Jointly or Qualifying Widow(er)	Married Filing Separately	Head of Household
10%	$0 – $8,350	$0 – $16,700	$0 – $8,350	$0 – $11,950
15%	$8,351 – $33,950	$16,701 – $67,900	$8,351 – $33,950	$11,951 – $45,500
25%	$33,951 – $82,250	$67,901 – $137,050	$33,951 – $68,525	$45,501 – $117,450
28%	$82,251 – $171,550	$137,051 – $208,850	$68,526 – $104,425	$117,451 – $190,200
33%	$171,551 – $372,950	$208,851 – $372,950	$104,426 – $186,475	$190,201 – $372,950
35%	$372,951+	$372,951+	$186,476+	$372,951+

PROBLEM: COMPUTING TAXES, CONT.

```
1.  if (status == 0) {
2.      // Compute tax for single filers
3.  }
4.  else if (status == 1) {
5.      // Compute tax for married file jointly
6.      // or qualifying widow(er)
7.  }
8.  else if (status == 2) {
9.      // Compute tax for married file separately
10. }
11. else if (status == 3) {
12.     // Compute tax for head of household
13. }
14. else {
15.     // Display wrong status
16. }
```

SWITCH STATEMENTS

```
1.   switch (status) {
2.     case 0:   //compute taxes for single filers;
3.             break;
4.     case 1:   //compute taxes for married file jointly;
5.             break;
6.     case 2:   //compute taxes for married file separately;
7.             break;
8.     case 3:   //compute taxes for head of household;
9.             break;
10.   default: System.out.println(
11.             "Errors: invalid status");
12.             System.exit(1);
13. }
```

SWITCH STATEMENT FLOW CHART
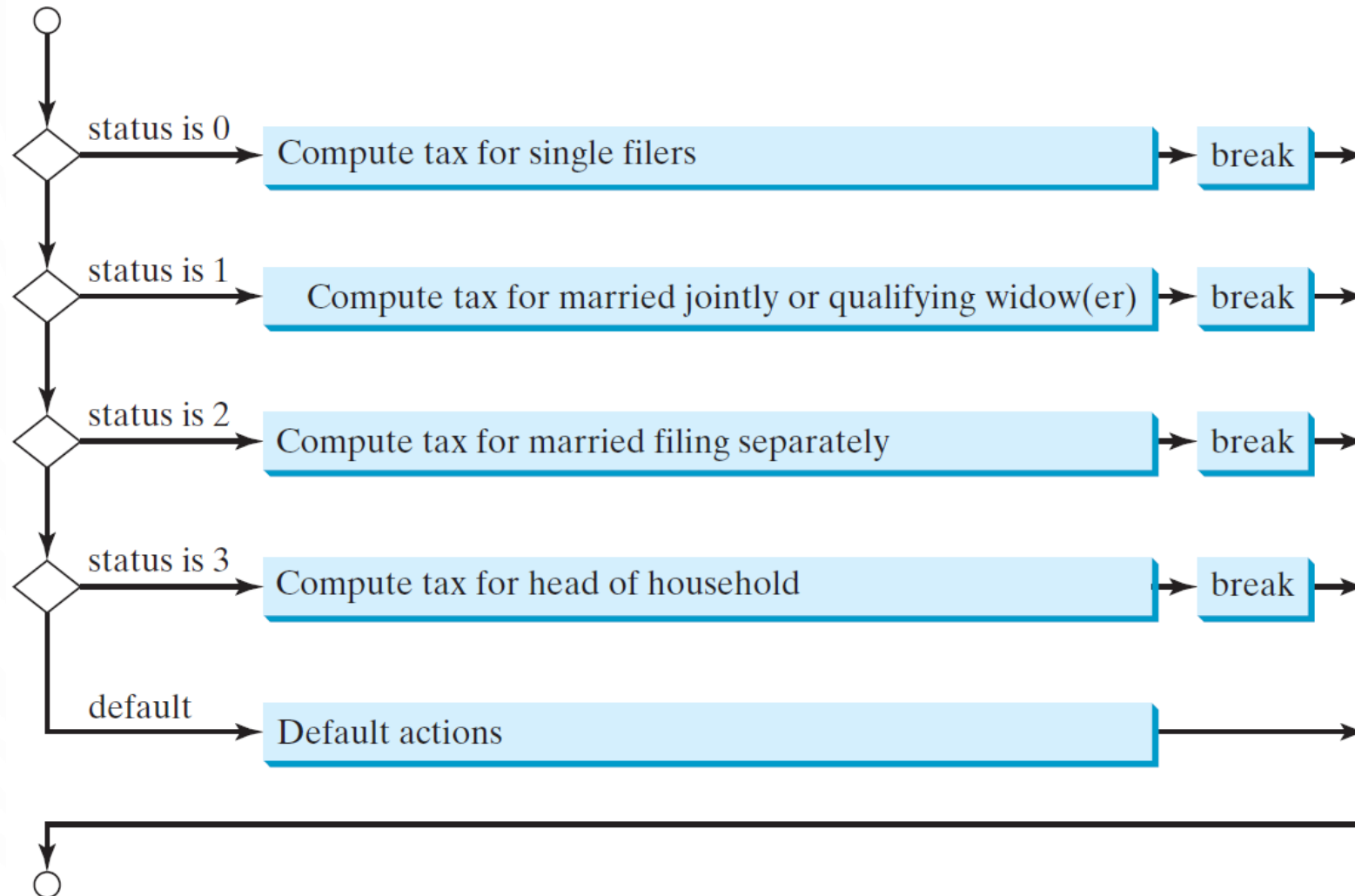
SWITCH STATEMENT RULES

- The switch-expression must yield a value of **char**, **byte**, **short**, or **int** type and must always be enclosed in parentheses.

- The value, ..., and valueN must have the same data type as the value of the switch-expression.

- The resulting statements in the case statement are executed when the value in the case statement matches the value of the switch-expression.

- Note that value1, ..., and valueN are constant expressions, meaning that they cannot contain variables in the expression, such as 1 + x.

```
1.    switch (switch-expression) {
2.       case value1: statement(s)1;
3.                    break;
4.       case value2: statement(s)2;
5.                    break;
6.       ...
7.       case valueN: statement(s)N;
8.                    break;
9.    default: statement(s)-for-default;
10. }
```

SWITCH STATEMENT RULES

- The keyword **break** is optional, but it should be used at the end of each case in order to terminate the remainder of the switch statement. **If the break statement is not present, the next case statement will be executed.**

- The **default** case, which is optional, can be used to perform actions when none of the specified cases matches the switch-expression.

- When the value in a case statement matches the value of the switch-expression, the statements starting from this case are executed until either a break statement or the end of the switch statement is reached.

```
1.   switch (switch-expression) {
2.     case value1: statement(s)1;
3.                  break;
4.     case value2: statement(s)2;
5.                  break;
6.     …
7.     case valueN: statement(s)N;
8.                  break;
9.     default: statement(s)-for-default;
10. }
```

CONDITIONAL EXPRESSIONS

```
1. if (x > 0)
2.    y = 1;
3. else
4.    y = -1;
```

- is equivalent to a special **ternary** operator

- `y = (x > 0) ? 1 : -1;`

- (boolean-expression) ? expression1 : expression2;